SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni preddiplomski studij računarstva

BROJILO PROMETA TEMELJENO NA RAČUNALNOJ OBRADI SLIKE

Završni rad

Filip Maras

Osijek, 2018.

SADRŽAJ

1. UVOD

Brojilo prometa je uređaj koji pomoću određenih senzora broji vozila. Brojilo prometa može se realizirati na više načina. To može biti pomoću induktivne petlje, ručnim brojanjem, magnetskim brojilima, obradom video slike i slično. U ovom radu naglasak je na brojanju prometa obradom digitalne slike. Ovakav način brojanja prometa je relativno nov i u povojima, ali s velikim razvojnim kapacitetom. Daljnjim razvojem i usavršavanjem algoritama za brojanje prometa postiže se velika točnost brojanja uz vrlo mala ulaganja. Problem kod primjerice brojača s induktivnom petljom je taj što se za postavljanje na određeno mjesto mora kopati sam kolnik što nije jeftin zahvat, a u konačnosti utječe na životni vijek samoga kolnika. Za brojanje prometa obradom digitalne slike potrebne su tri stvari: računalo, web-kamera i program koji će brojati promet. Takav sustav je lako prenosiv, a podatke možemo vidjeti u realnom vremenu. U ovom radu bit će opisani alati koji su korišteni prilikom izrade takvog brojača, bit će objašnjen algoritam pomoću kojeg se broji promet i sve dodatne korake koji se moraju napraviti kako bi se mogao realizirati jedan takav brojač. Također, bit će prikazani eksperimentalni rezultati brojanja prometa uz uspješnost prikazanu u postocima.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je realizacija brojača prometa na temelju obrade digitalne slike. Potrebno je da program u stvarnom vremenu broji vozila uz pogrešku svedenu na minimum.

Nakon što program bude napisan, biti će testiran na testnim video isječcima uz prikaz uspješnosti za svaki.

2. KONCEPT SUSTAVA I TEHNOLOGIJE

U ovom poglavlju bit će opisan osnovni princip rada brojila prometa temeljeno na obradi slike. Također će biti spomenuti C# i Emgu CV koji su korišteni pri realizaciji programskog dijela završnog rada.

2.1. Brojilo prometa (obradom slike)

Video se sastoji od sličica (engl. *frame*). Broj sličica ovisi o vrsti i kvaliteti kamere. Inače se radi o 24 do 30 sličica u sekundi (engl. *frame rate*). Naravno, što je više sličica video je veće kvalitete i izgleda glađe. Na kvalitetu utječe i rezolucija videa. Na videu veće rezolucije se puno bolje vide detalji nego na videu manje rezolucije. Nažalost, video isječci veće kvalitete su i puno teži za obradu, tj. programu treba puno više vremena kako bi napravio istu stvar kao na videu lošije kvalitete. Kako je u slučaju brojanja vozila nevažna kvaliteta videa, ali važno je da program broji u realnom vremenu bez gubljenja podataka, koriste se video isječci smanjene kvalitete.

Kako bi se uopće mogao brojati promet obradom slike potrebno je imati kameru koja će snimati vozila u prolasku te računalo s programom koje će obrađivati, tj. brojati vozila. Za svrhu ovoga rada korištena je web kamera pomoću koje se u realnom vremenu broje prolasci vozila. Također su zbog jednostavnosti korišteni i unaprijed snimljeni video isječci.

Kada se radi s videom koji je uživo, program dohvaća sličice izravno s kamere. Zbog toga se mora paziti da se svaka sličica ne obrađuje predugo jer ako program ne dohvati sljedeću sličicu s kamere, ona se gubi što u konačnici može dovesti do netočnih rezultata.

U slučaju kada se radi s prethodno snimljenim videom, neće doći do gubitaka podataka jer je video spremljen na nekom od medija za pohranu i program dohvaća sljedeću sličicu tek kada obradi prethodnu. To znači da će se obraditi svaka sličica. Međutim, mogu se imitirati stvarni uvjeti tako da se dohvaća svaka druga ili treća sličica, ovisno o brzini izvršavanja programa.

2.2. C# i Emgu CV

C# je objektno orijentirani programski jezik koji je razvila tvrtka Microsoft. Nastao je 2000. godine u sklopu .NET platforme. To je jedan od najrasprostranjenijih programskih jezika. Koristi se u izradi aplikacija za Windows operacijski sustav, web aplikacija, mobilnih aplikacija, u izradi igara i drugo. Sintaksa C#-a je slična C-u, C++-u i Javi. Naredbe završavaju točka-zarezom, zagrade se koriste za grupiranje naredbi. Naredbe se uglavnom grupiraju u funkcije,

funkcije u klase, a klase u imenike (engl. *Namespace*). Također je potrebno za sve varijable navesti njen tip. C# je jezik kojim se mogu pisati vrlo velike i komplicirane aplikacije. Ono što to omogućava su razne značajke. Neke od njih su: skupljanje smeća koje automatski oslobađa memoriju koju zauzimaju nekorišteni objekti, zatim rukovanje iznimkama koje omogućuje lako i brzo otkrivanje i otklanjanje grešaka, te mnoge druge.

U C#-u postoje dvije vrste tipova podataka. To su vrijednosni tipovi i pokazivači. Vrijednosni tipovi sadrže podatak, a pokazivači adresu podatka. Vrijednosni tipovi se dalje dijele na jednostavne, *enum, null i* strukture. Pokazivači se dijele na klase (razrede), sučelja, polja i tipovi delegati. Klase su osnovni tip podataka u C#. Klasu se može zamisliti kao model pomoću kojeg se stvaraju objekti. Klasa se sastoji od dva dijela. Prvi dio sadrži sve atribute klase, njezino ime i drugo. Drugi dio sadrži metode pomoću kojih pristupamo atributima kako bi ih promijenili, ispisali ili koristili u nekoj drugoj metodi. Klase također omogućuju nasljeđivanje i polimorfizam. Nasljeđivanje nam omogućuje da jedna ili više klasa od klase roditelja naslijedi sve ili samo dio atributa i metoda klase roditelj. To nam omogućuje da sve karakteristike koje su zajedničke za određeni problem ili fizički objekt stavimo u klasu roditelj. Ostale klase će naslijediti te karakteristike, a uz to će imati i neke druge karakteristike koje ih onda pobliže definiraju.[1]

Emgu CV je .NET biblioteka koja sadrži metode za obradu slike. U suštini Emgu CV je Open CV za .NET platformu. To znači da sve metode open CV-a mogu biti pozvane jezicima koji su kompatibilni sa .NET platformom. [2][3]

Biblioteke korištene u realizaciji ovoga rada su: Emgu.CV, Emgu.CV.UI, Emgu.CV.Structure.

Emgu.CV biblioteka sadrži Open.CV metode za obradu slike. Emgu.CV.UI sadrži objekte za prikaz slike (ImageBox i slično), a Emgu.CV.Structure sadrži Open.CV strukture.

U ovom radu korišten je samo mali dio mogućnosti koje ima Emgu CV. Emgu CV nam daje na raspolaganje više od 500 metoda za obradu digitalne slike. Neke od tih metoda su: metoda za detekciju registracijskih pločica, detekcija lica, detekcija oka, oblika i druge.

3. ALGORITAM ZA BROJANJE PROMETA

U ovom poglavlju objašnjene su sve metode koje su korištene pri predobradi slike. U potpoglavlju 3.2. objašnjen je algoritam kojim se broje vozila u prolasku.

3.1. Predobrada slike

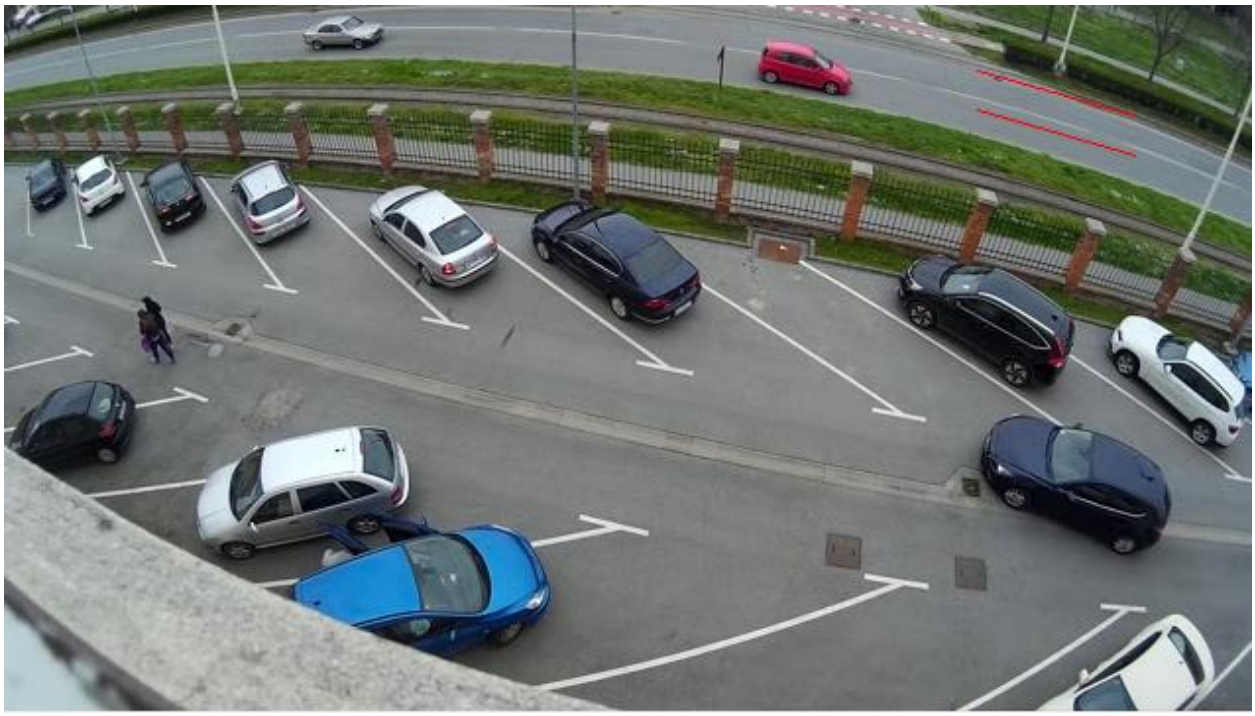
Kako bi bilo moguće prebrojati vozila koja prolaze, potrebno je napraviti nekoliko koraka predobrade slike. Prvo je potrebno sliku u boji koja dolazi s kamere ili postoji kao video u memoriji i sprema se u varijablu *_capture2* pretvoriti u crno-bijelu sliku i spremiti ju u novu varijablu *GrayImage*. Sada se uvodi nova varijabla oldImage. Za sam početak videa potrebno je provjeriti je li oldImage *null*, ako je, u njega se sprema *GrayImage*. Sljedeći korak je razlika između *GrayImage*-a i *oldImage*-a. Razlika se sprema u *resultImage2*. Nadalje se sve operacije rade nad *resultImage2*. Prvi sljedeći korak je pretvaranje *resultImage2* u binarnu sliku. To se vrši funkcijom *ThresholdBinary* na sljedeći način (slika 3.1.):

```
resultImage2._ThresholdBinary(new Gray(20), new Gray(255));
```

Slika 3.1. Funkcija ThresholdBinary

ThresholdBinary funkcionira tako da provjerava intenzitet točaka i ako je isti veći od zadane vrijednosti Thresh-a (u ovom slučaju 20), intenzitet se stavlja na maksimalnu vrijednost, što je 255 u našem slučaju. Sve vrijednosti intenziteta točke manje od zadane vrijednosti se stavljaju u nulu. Važno je naglasiti da 0 predstavlja crnu boju, a 255 bijelu. Na ovaj način se dobiva čisto crno-bijela slika (binarna) bez nijansi između[4]. U nastavku je prikazana usporedba slike koja je spremljena u *resultImage2* (slika 3.2.) i njenog originala (slika 3.3.).

Slika 3.2. Binarizirana slika



Slika 3.3. Originalna slika prije binarizacije

Morfologija predstavlja jednostavne operacije koje se temelje na oblicima i karakteristikama slike. Najčešće se izvodi na binarnim slikama. Postoje dvije osnovne morfološke operacije, a to su erozija i dilacija [6].

Erozija se ponaša kao prirodna erozija. Granice objekta će erodirati i objekt će postati tanji, a princip je jednostavan. Kako se radi nad binarnim slikama, prolaskom kroz sliku po pikselima, svi pikseli koji su bijeli, a oko sebe nemaju samo bijele piksele bit će erodirani, odnosno postat će crni. Dilacija je suprotna eroziji. Prolaskom kroz sliku po pikselima, svi pikseli koji su crni, a oko sebe imaju bijele piksele, postaju bijeli. Za rezultat naš objekt postaje deblji. Erozijom možemo ukloniti šumove, a dilacijom spojiti prekinute dijelove.

Erozija i dilacija nisu jedine morfološke operacije. Postoje još mnoge, a neke od tih su otvaranje i zatvaranje. U realizaciji ovog rada korišteno je otvaranje.

Otvaranje (slika 3.5.) je operacija koja koristi eroziju i dilaciju. Prvo se erozijom uklanjaju šumovi, a zatim dilacijom spajamo dijelove koji nisu spojeni. Otvaranje je operacija koja se najčešće koristi za uklanjanje šumova, dok se zatvaranje (slika 3.4.) koristi za popunjavanje rupa unutar objekta [5].

Slika 3.4. Primjer zatvaranja [5]

Slika 3.5. Primjer otvaranja [5]

3.2. Brojanje vozila

Kako je naš početni video prošao kroz predobradu, sada je spreman za najbitniji dio, tj. brojanje prometa. Prvo je potrebno odrediti tri točke po traci koje će provjeravati boju. Znajući da je sve statično crno, a dinamično bijelo, auto koji je u pokretu će biti bijel. Tako, kada auto prolazi preko prethodno odabranih točaka, one postaju bijele. Sada kad osnovna detekcija pokreta postoji, potrebno je odrediti smjer kretanja. Znajući da na određenoj prometnici auti jednom trakom dolaze iz jednog smjera, a drugom iz suprotnog, jednostavnom provjerom redoslijeda kojim točke postaju bijele, program određuje smjer vozila i broji ga.

U nastavku je prikazan kod za traku kojom vozila nadolaze s desne na lijevu stranu (slika 3.6.).

```
if (AR > 0 && flag1 == 0 && flag2 == 0 && flag3 == 0) flag1 = 1;
if (A > 0 && flag1 == 1 && flag2 == 0 && flag3 == 0) flag2 = 1;
if (AL > 0 && flag1 == 1 && flag2 == 1 && flag3 == 0) flag3 = 1;
if (flag3 == 1 && flag2==1 && flag3==1)
{
    CounterL();
    flag1 = 0;
    flag2 = 0;
    flag3 = 0;
}
```

Slika 3.6. Kod koji prikazuje prepoznavanje kretanja sa desna na lijevo

U varijable tipa byte *A*, *AR*, *AL* su spremljene vrijednosti točaka [0,255]. Kako se radi o binarnoj slici ta vrijednost može biti 0 ili 255. Sada je samo potrebno redom od desno na lijevo provjeravati je li vrijednost točke veća od nule, odnosno pojavljuje li se na točki bijela boja. Dakle, ako je vrijednost *AR* veća od nule pomoćna varijable *flag1*, *flag2* i *flag3* su jednake nuli, točka je bijela i to znači da vozilo prolazi preko te točke. Sada pomoćnu varijablu *flag1* tipa int postavljamo u jedan. Sljedeći korak je provjera srednje točke čija je vrijednost spremljena u varijablu *A*. Prvo se provjerava je li vrijednost *AR* veća od nule, je li *flag1* jednaka jedan što znači da je prva točka koja je bila bijela ona čija je vrijednost spremljena u *AR* i jesu li *flag2* i *flag3* jednake nuli. Ako su ti uvjeti zadovoljeni, *flag2* postavljamo u jedan. Nadalje, provjerava se i zadnja točka u redu, ona čija je vrijednost spremljena u varijablu *AL*. Kao i za *A* i *AR*, prvo se provjerava je li vrijednost točke veća od nule. *Flag1* i *flag2* moraju imati vrijednost 1 što znači da su te točke prve pobijelile, da je smjer kretanja ispravan i da nije došlo do slučajne pogreške. Također flag3 u provjeri mora biti jednak nuli. Ako su ti uvjeti zadovoljeni, *flag3* postavljamo u jedan. Zadnja provjera se vrši

nad varijablom *flag3*, ako je ona jednaka jedan tada se poziva metoda *CounterL* koja će povećati za jedan varijablu *BrL* koja predstavlja broj vozila koja su prošla iz smjera desno na lijevo. Sada je samo potrebno sve tri flag varijable postaviti u nulu kako bi bile spremne za sljedeći prolazak vozila.

Za traku kojom vozila nadolaze s lijeve na desnu stranu problem je riješen istom logikom. Jedina razlika je što se prvo provjerava lijeva točka, zatim srednja i na kraju desna. Kod je u nastavku (slika 3.7.).

```
if (BL > 0 && flag4==0 && flag5==0 && flag6==0) flag4 = 1;
if (B > 0 && flag4 == 1 && flag5 == 0 && flag6 == 0) flag5 = 1;
if (BR > 0 && flag4 == 1 && flag5 == 1 && flag6 == 0) flag6 = 1;
if (flag6 == 1 && flag4==1 && flag5==1)
{
    CounterR();
    flag4 = 0;
    flag5 = 0;
    flag6 = 0;

}
```

Slika 3.7. Kod koji prikazuje prepoznavanje kretanja sa lijeva na desno

Varijabla *BL* predstavlja vrijednost točke s lijeve strane i to je ujedno točka na koju vozilo prvo nadolazi. Varijabla *B* predstavlja vrijednost srednje točke, a *BR* točke s lijeve strane.

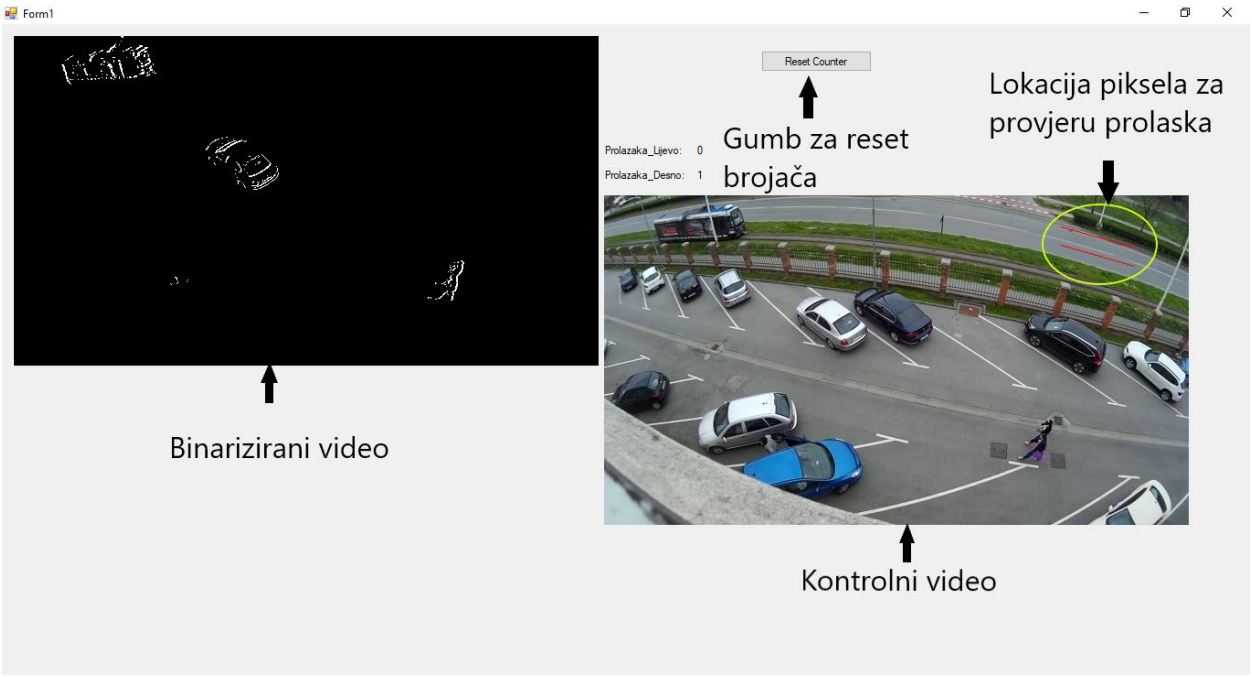
CounterR je metoda koja povećava varijablu brR za jedan svaki put kada su zadovoljeni uvjeti gore navedeni. Varijabla *BrR* predstavlja broj vozila koja su prošla s lijeve na desnu stranu cestom.

4. EKSPERIMENTALNI REZULTATI

Uz prikaz eksperimentalnih rezultata i usporedbu brojanja programom s brojanjem čovjeka, ovo poglavlje još sadrži opis grafičkog sučelja i opis skupljanja podataka koji su potrebni za provedbu eksperimentalnog dijela završnog rada.

4.1. Grafičko sučelje

Grafičko sučelje je dio programa koji korisniku omogućuje komunikaciju s računalom, odnosno omogućuje mu iskorištavanje funkcionalnosti odabranog programa. Grafičko sučelje brojača prometa se sastoji od sedam elemenata, a to su: dva okvira slike (*ImageBox*), četiri naljepnice (*Label*) i jednog gumba (*Button*). Okvir slike (*ImageBox*) je klasa iz biblioteke Emgu.CV.UI koji omogućuje prikaz i slike i manipulaciju iste. Naljepnica (*Label*) je klasa iz biblioteke System.Windows.Forms koja omogućuje označavanje ili ispis teksta iz programa. Gumb (*Button*) je također klasa sadržana u biblioteci System.Windows.Forms koja na događaj klika na gumb odrađuje ranije definirani posao. U nastavku se može vidjeti prikaz grafičkog sučelja brojača prometa (slika 4.1.).



Slika 4.1. Grafičko sučelje

Kao što se vidi, jedan okvir slike prikazuje binarizirani video, a drugi kontrolni video. Binarizirani video je onaj koji se obrađuje, tu se broji promet. Kontrolni video služi samo krajnjem korisniku kako bi vidio što točno snima s kamerom te da zna gdje se nalaze točke kojima

prepoznajemo kretanje (dva crvena pravca prikazana na kontrolnom videu. Od četiri naljepnice dvije nam služe za opis. To su naljepnica Prolazaka_Lijevo i naljepnica Prolazaka_Desno. Te dvije naljepnice se ne mijenjaju, za razliku od druge dvije koje pokazuju broj automobila koji su prošli u određenom smjeru. Ako je potrebno brojače staviti na nulu, koristi se gumb Reset Counter (slika 4.2.).

```
private void Reset_counter_Click(object sender, EventArgs e)
{
    BrL = 0;
    BrR = 0;
}
```

Slika 4.2. Metoda za reset brojača

BrL i *BrR* su varijable tipa int koje svakim prolaskom vozila ulijevo (*BrL*) ili udesno (*BrR*) povećavamo za jedan dvjema metodama *CounterL* i *CounterR* (slika 4.3.) koje su ranije spomenute.

```
private void CounterL()
{
    BrL++;
}
private void CounterR()
{
    BrR++;
}
```

Slika 4.3. Metode brojača

4.2. Prikupljanje ulaznih podataka

Za potrebe ovog završnog rada korištena je web kamera marke Logitech. Promet je sniman na tri lokacije. S trećeg kata zgrade Fakulteta elektrotehnike, računarstva i informacijskih tehnologija na Kampusu (slika 4.5.), sa zgrade u Ulici Grada Vukovara u Čepinu (slika 4.7.) i s nadvožnjaka iznad autoceste Slavonika nakon izlaza Čepin (slika 4.6.). Bilo je bitno da se promet snima s visine kako vozila u traci bližoj kameri ne bi zaklanjali vozila u traci koja je dalja od kamere. Zbog toga nije moguće snimati prometnicu s boka u razini prometnice. Prometnica koja je snimana sa zgrade fakulteta snimana je s boka (vozila prolaze ulijevo i udesno), te je na primjeru tog videa objašnjen cijeli rad. Slavonika je snimana od gore, pa tako vozila idu gore dolje na videu.

Za ovaj tip videa potrebno je samo promijeniti lokaciju piksela koji detektiraju kako bi se nalazili na trakama.

Za lakše korištenje lokacija piksela se određuje unutar samog programa pritiskom miša na mjesto na slici gdje želimo da naši detektirajući pikseli budu. Taj problem je riješen implementacijom događaja (event) Mouse_Click na kontrolni video uz switch-case grananje (slika 4.4.).

Svakim pritiskom miša na kontrolni video spremaju se koordinate gdje se miš nalazi u tom trenutku i spremaju se u varijable X i Y. Također, svaki pritisak povećava brojač za jedan, što utječe na switch-case grananje koje ovisno o slučaju sprema vrijednosti iz varijabli X i Y u pripadajuće varijable koje se koriste za detekciju. Za pravilno postavljanje lokacija, potrebno je pratiti naputak koji se nalazi u samoj aplikaciji i koji govori o redoslijedu kojim se lokacije moraju postavljati.

```
private void ibVideo_MouseClick(object sender, MouseEventArgs e)
{
    X = e.X;
    Y = e.Y;
    brojac++;
    ispisxy.Text = X.ToString() +" "+ Y.ToString() +" "+brojac;

    switch(brojac)
    {
        case 1:
            left_oneX = X;
            left_oneY = Y;
            break;
        case 2:
            middle_oneX = X;
            middle_oneY = Y;
            break;
        case 3:
            right_oneX = X;
            right_oneY = Y;
            break;
        case 4:
            right_twoX = X;
            right_twoY = Y;
            break;
        case 5:
            middle_twoX = X;
            middle_twoY = Y;
            break;
        case 6:
            left_twoX = X;
            left_twoY = Y;
            break;
        default:
            break;
    }
}
```

Slika 4.4. Postavljanje lokacije piksela za prepoznavanje kretanja

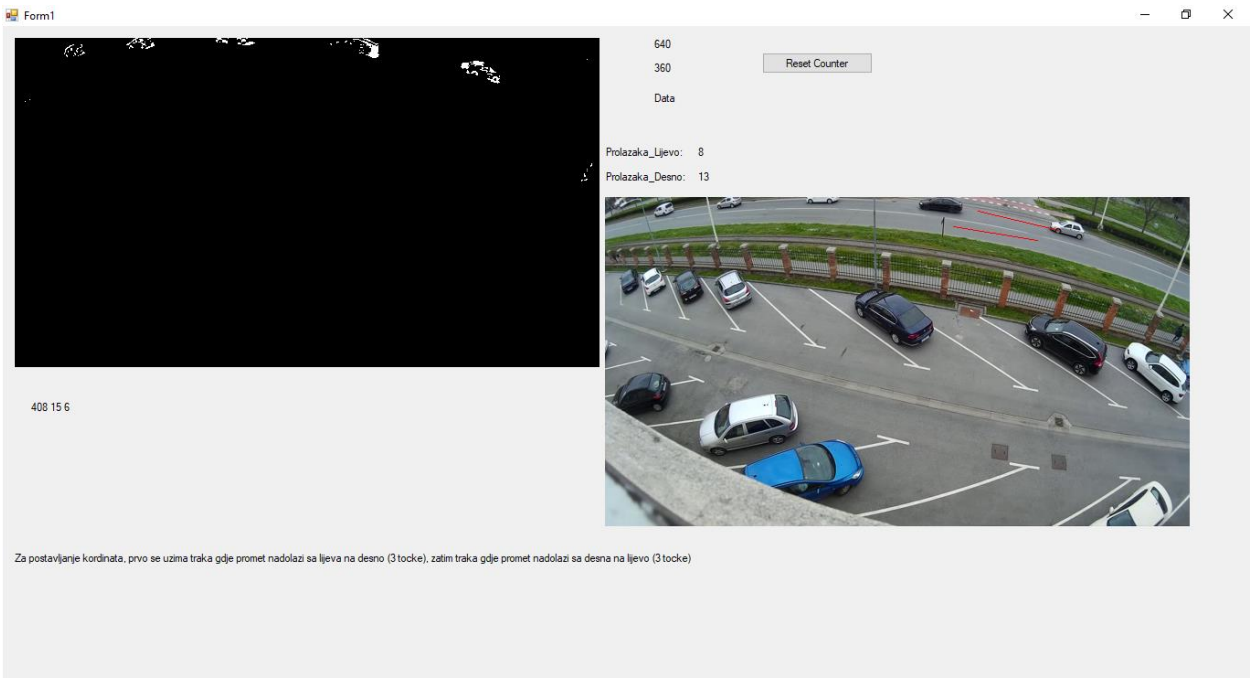
Slika 4.5. Snimka sa zgrade na Kampusu

Slika 4.6. Snimka Slavonike

Slika 4.7. Ulica Grada Vukovara, Čepin

4.3. Uspješnost

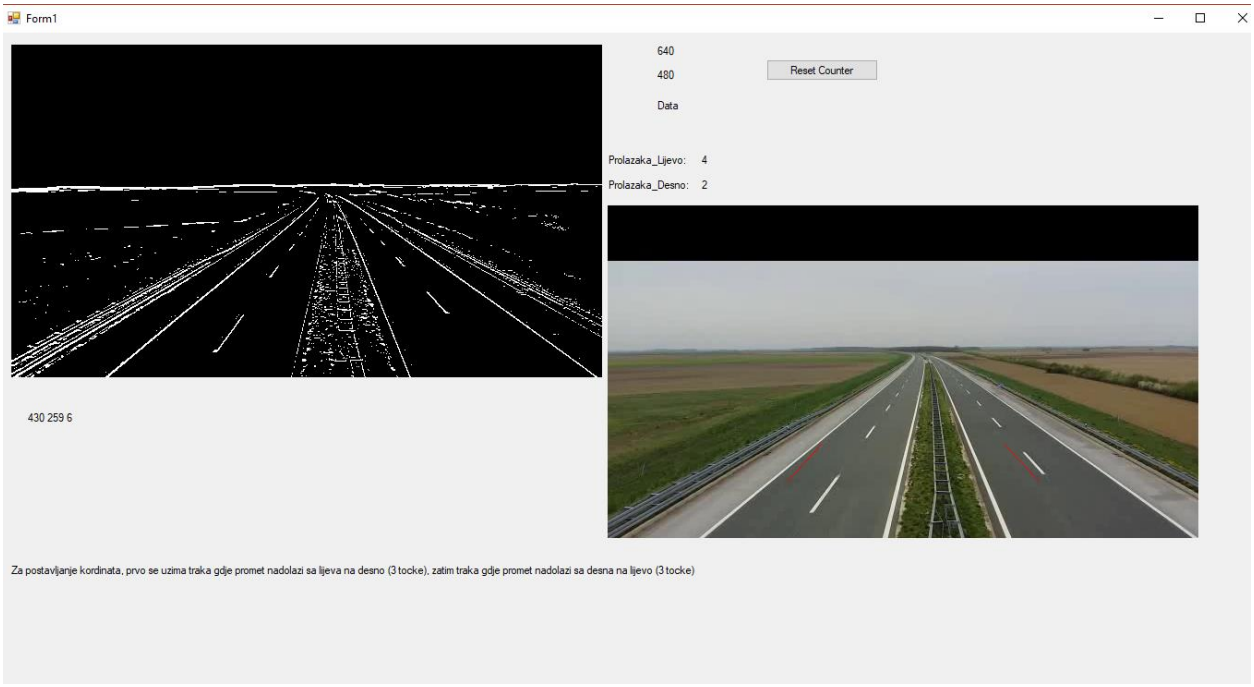
U ovom potpoglavlju biti će prikazane eksperimentalne snimke sa rezultatima brojanja programa i čovjeka (slike 4.8. do 4.14.). Svi dobiveni rezultati biti će objašnjeni i prikazani postotku uspješnosti u tablicama (tablice 4.1. do 4.7.). Uz pretpostavku da će čovjek točno prebrojati vozila, naš rezultat će biti omjer vozila prebrojanih programom te broja vozila koje je prebrojao čovjek.



Slika 4.8. Prvo eksperimentalno snimanje sa zgrade fakulteta na Kampusu

Tablica 4.1. Prvo eksperimentalno snimanje sa zgrade fakulteta na Kampusu

Lokacija		Fakultet elektrotehnike računarstva i informacijskih tehnologija Osijek, zgrada na Kampusu	
Trajanje snimanja		1 minuta	
Uspješnost	Prometna traka s lijeva na desno	Broj vozila koja je prebrojao program	13
		Broj vozila koja je prebrojao čovjek	13
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	0
	Prometna traka s desna na lijevo	Broj vozila koja je prebrojao program	8
		Broj vozila koja je prebrojao čovjek	8
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	0

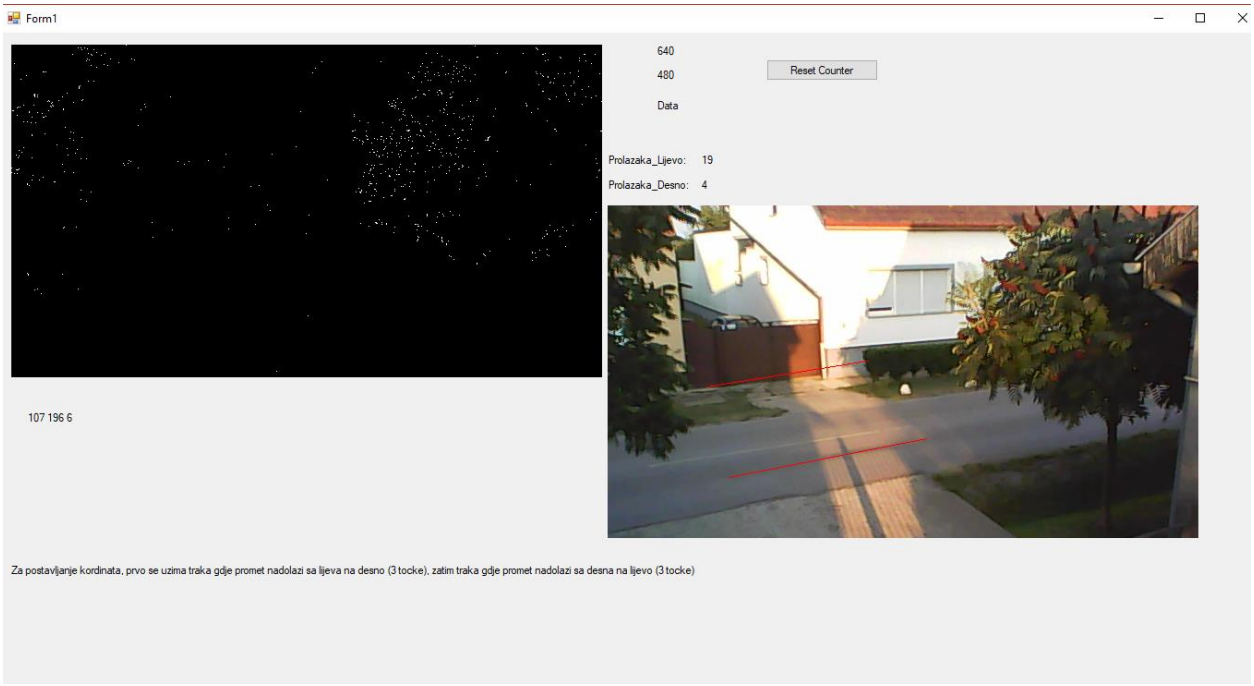


Slika 4.9. Drugo eksperimentalno snimanje, autocesta Slavonika

Tablica 4.2. Drugo eksperimentalno snimanje, autocesta Slavonika

Lokacija		nadvožnjak u Čepinu iznad autoceste Slavonika	
Trajanje snimanja		2 minute	
Uspješnost	Prometna traka od gore prema dolje	Broj vozila koja je prebrojao program	2
		Broj vozila koja je prebrojao čovjek	1
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	1
	Prometna traka od dolje prema gore	Broj vozila koja je prebrojao program	4
		Broj vozila koja je prebrojao čovjek	4
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	0

Program je prebrojao sva vozila, ali jedno vozilo je brojao dva puta.

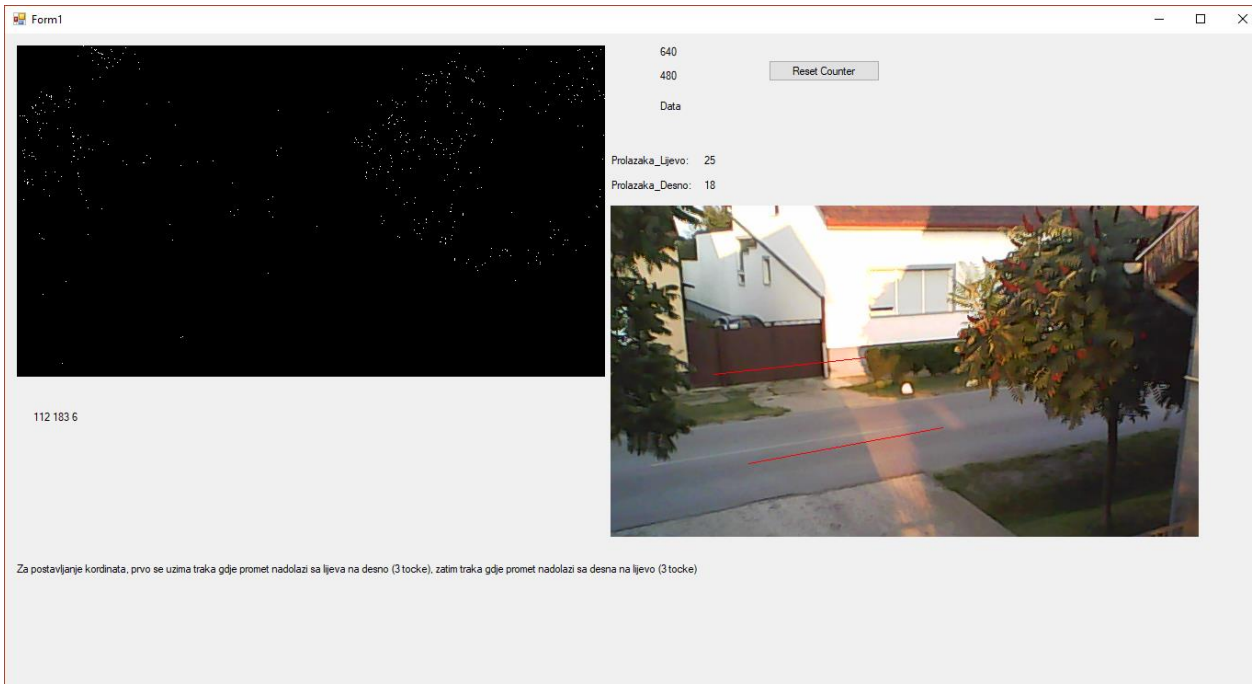


Slika 4.10. Treće eksperimentalno snimanje, Čepin

Tablica 4.3. Treće eksperimentalno snimanje, Čepin

Lokacija		Čepin, Ulica Grada Vukovara	
Trajanje snimanja		15 minuta	
Uspješnost	Prometna traka s lijeva na desno	Broj vozila koja je prebrojao program	4
		Broj vozila koja je prebrojao čovjek	4
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	0
	Prometna traka s desna na lijevo	Broj vozila koja je prebrojao program	19
		Broj vozila koja je prebrojao čovjek	16
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	3

Kao što se vidi, program je točno brojao promet za traku gdje promet dolazi s lijeva na desno. Međutim, izbrojao je više prolazaka za traku gdje promet dolazi s desna na lijevo. To se dogodilo jer je kamera postavljena tako da gleda okomito na promet i nije na dovoljnoj visini, pa se vozila preklapaju na slici što dovodi do krivih rezultata.

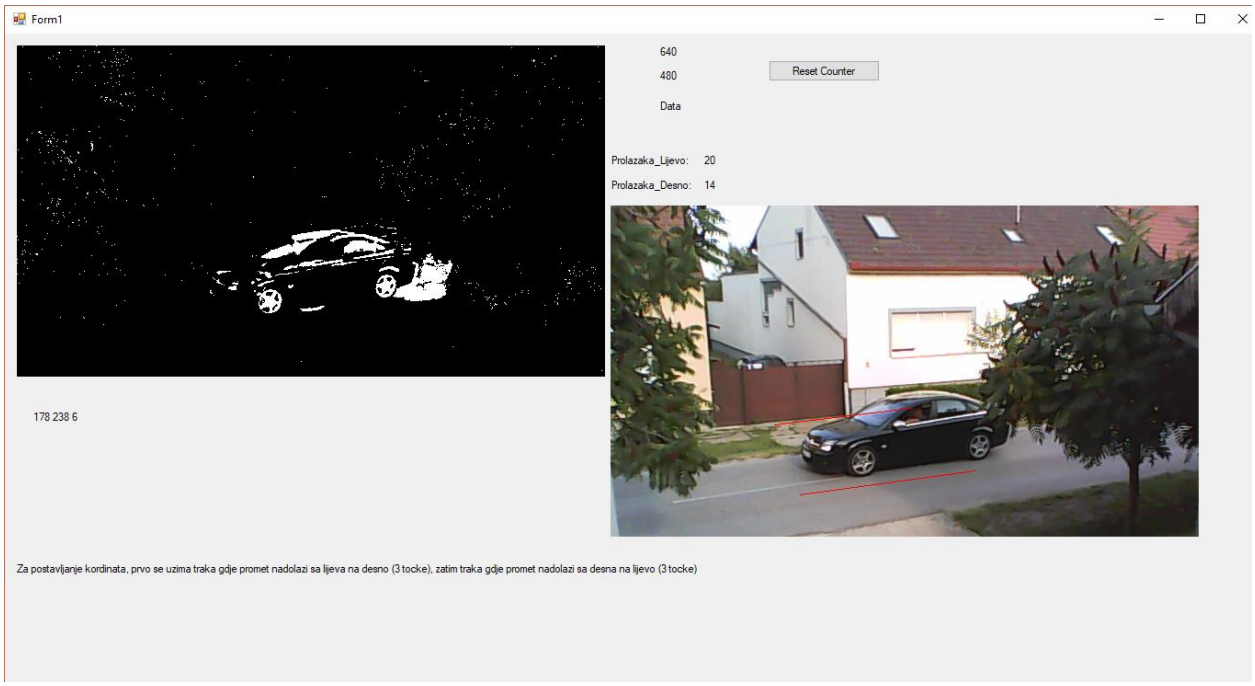


Slika 4.11. Četvrto eksperimentalno snimanje, Čepin

Tablica 4.4. Četvrto eksperimentalno snimanje, Čepin

Lokacija		Čepin, Ulica Grada Vukovara	
Trajanje snimanja		30 minuta	
Uspješnost	Prometna traka s lijeva na desno	Broj vozila koja je prebrojao program	18
		Broj vozila koja je prebrojao čovjek	18
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	0
	Prometna traka s desna na lijevo	Broj vozila koja je prebrojao program	25
		Broj vozila koja je prebrojao čovjek	24
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	1

U ovom se primjeru također vidi da je program izbrojao jedno vozilo više nego čovjek. Međutim, zbog malog namještanja mjesta prepoznavanja kretanja (crveni pravci) greška je manja nego u prethodnom primjeru.

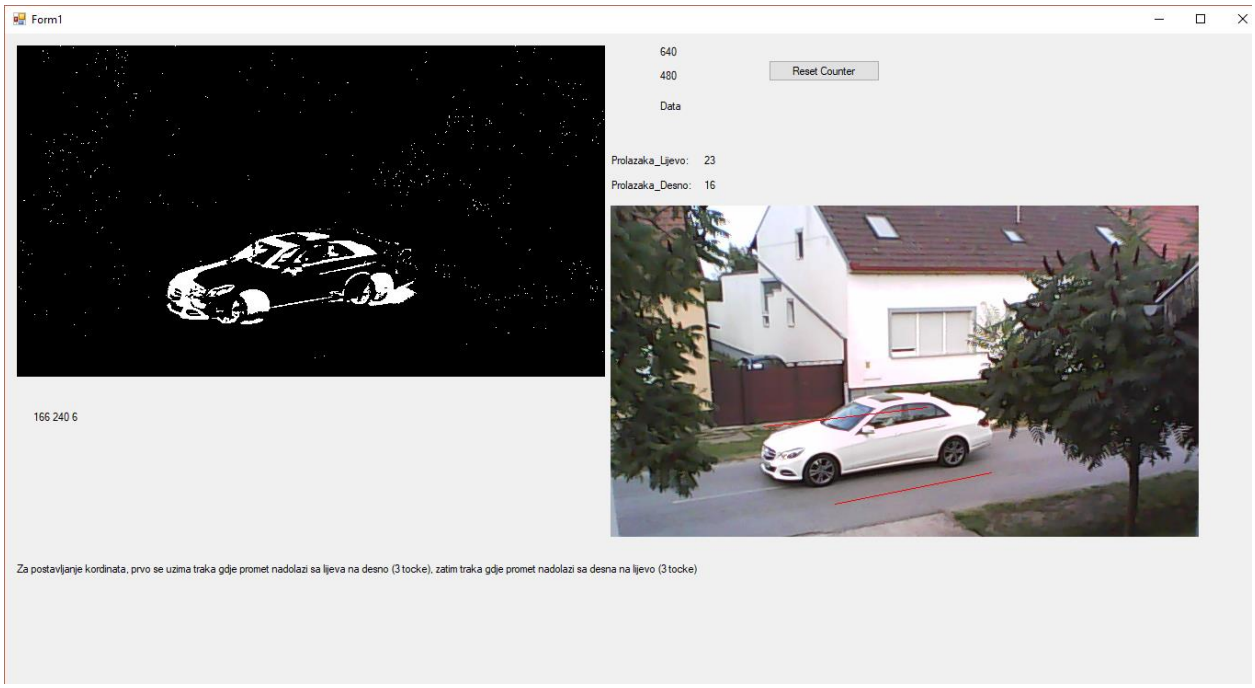


4.12. Peto eksperimentalno snimanje, Čepin

Tablica 4.5. Peto eksperimentalno snimanje, Čepin

Lokacija		Čepin, Ulica Grada Vukovara	
Trajanje snimanja		15 minuta	
Uspješnost	Prometna traka s lijeva na desno	Broj vozila koja je prebrojao program	14
		Broj vozila koja je prebrojao čovjek	13
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	1
	Prometna traka s desna na lijevo	Broj vozila koja je prebrojao program	20
		Broj vozila koja je prebrojao čovjek	19
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	1

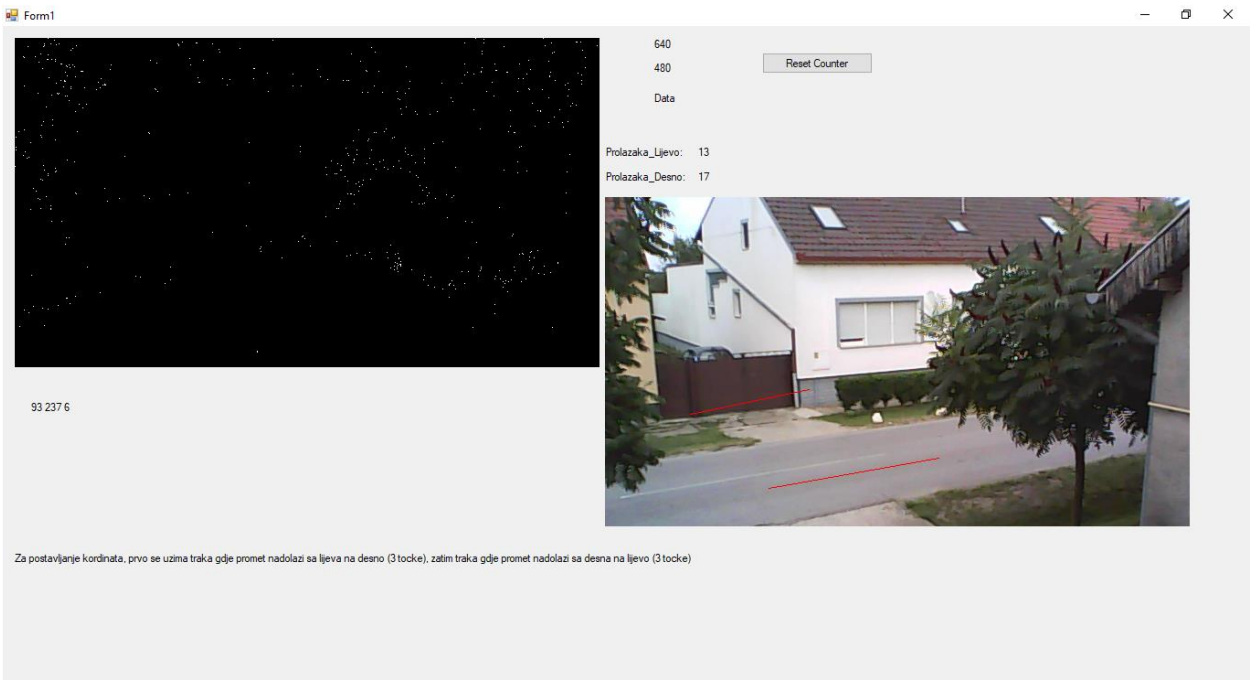
I u ovom primjeru se vidi da je program prebrojao nekoliko vozila više nego čovjek. Razlog je isti. Zbog kuta i visine snimanja dogodila su se preklapanja na slici zbog kojih je program prepoznao prolazak iako ga u stvarnosti nije bilo.



4.13. Šesto eksperimentalno snimanje, Čepin

Tablica 4.6. Šesto eksperimentalno snimanje, Čepin

Lokacija		Čepin, Ulica Grada Vukovara	
Trajanje snimanja		20 minuta	
Uspješnost	Prometna traka s lijeva na desno	Broj vozila koja je prebrojao program	16
		Broj vozila koja je prebrojao čovjek	14
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	2
	Prometna traka s desna na lijevo	Broj vozila koja je prebrojao program	23
		Broj vozila koja je prebrojao čovjek	22
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	1



4.14. Sedmo eksperimentalno snimanje, Čepin

Tablica 4.7. Sedmo eksperimentalno snimanje, Čepin

Lokacija		Čepin, Ulica Grada Vukovara	
Trajanje snimanja		20 minuta	
Uspješnost	Prometna traka s lijeva na desno	Broj vozila koja je prebrojao program	17
		Broj vozila koja je prebrojao čovjek	16
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	1
	Prometna traka s desna na lijevo	Broj vozila koja je prebrojao program	13
		Broj vozila koja je prebrojao čovjek	9
		Uspješnost brojanja programom	100%
		Broj slučajnih pogrešnih detekcija	4

5. ZAKLJUČAK

Brojilo prometa realizirano digitalnom obradom slike ima velike prednosti u odnosu na slične sustave kompliciranije izvedbe. Za realizaciju nam je potrebno računalo, kamera i program. A činjenica da se većina prometnica već snima nam jako olakšava posao jer je potrebno samo taj isti video pustiti kroz program koji će brojati vozila. To u konačnosti znatno utječe na smanjenje troškova oko brojanja vozila jer se koristi postojeći sustav kamera i ne zahtijeva dodatne radove na prometnicama. Program koji je realiziran u svrhu ovog završnog rada ne broji apsolutno točno, ali pokazuje tehnologije i tehnike koje se mogu koristiti u svrhu izrade komercijalnog brojača prometa digitalnom obradom slike. Bitno je naglasiti da sam položaj kamere može jako utjecati na podatke koje dobivamo u konačnici. Idealno bi bilo kada bi kamera bila iznad prometnice tako da se vozila međusobno ne zaklanjaju. U situacijama kada kamera bočno snima prometnicu, bilo bi dobro da je na većoj visini kako ne bi vozila koja prometuju trakom bliže kameri zaklanjala vozila koja prometuju trakom koja je dalje od kamere. Takav slučaj se vidi na snimkama u Ulici Grada Vukovara gdje se kamera ne nalazi dovoljno visoko te iz tok razloga i točnost podataka nije zadovoljavajuća. Probleme stvaraju kamioni i traktori koje zbog prostora između prikolice i vozila program broji kao više vozila. Ti slučajevi su razlog zašto je program brojao više vozila nego čovjek, jer čovjek prepoznaje kamion s prikolicom kao cjelinu dok program ne. U dosadašnjim testnim snimanjima program je dovoljno točno brojao promet. Glavni nedostatak bi bio upravo taj što povremeno broji više vozila nego zaista prođe.

LITERATURA

[1] Microsoft .Net documentation,

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction

Posjećeno: 20.6.2018.

[2] Emgu.CV main page,

http://www.emgu.com/wiki/index.php/Main_Page

Posjećeno: 20.6.2018.

[3] OpenCV main page,

https://opencv.org/about.html

Posjećeno: 20.6.2018.

[4] OpenCV documentation, Basic Thresholding Operations

https://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html

Posjećeno: 20.6.2018.

[5] OpenCV documentation, Morphological Transformations

https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html

Posjećeno: 20.6.2018.

[6] Rafael C. Gonzalez, Richard Eugene Woods, Digital Image Processing, Prentice Hall, 2008

SAŽETAK

Naslov: Brojilo prometa temeljeno na računalnoj obradi slike

Cilj ovog završnog rada je upoznavanje s novim tehnologijama obrade slike. Također, ukratko su predstavljene osnove programskog jezika C# koji je jedan od najkorištenijih programskih jezika danas. Uz C# korištena je i Emgu.CV biblioteka koja sadrži metode za obradu digitalne slike. Primjenom tih tehnologija realiziran je program koji obradom slike broji vozila u prometu. Promet se broji za svaku traku posebno što u konačnici omogućuje bolje interpretiranje podataka. Ovim programom smo ujedno i dali uvid u neke od mogućnosti digitalne obrade slike.

Ključne riječi: C#, Emgu.CV, brojač prometa, digitalna obrada slike

ABSTRACT

Title: Traffic Vehicle Counter Based on Digital Image Processing

The goal of this final paper is to introduce new technology in digital image processing. Also, we got to know a program language C# which is one of the most used programing languages today. Along with C# Emgu CV which contains methods for digital image processing was used. Using those technologies, a program based on image processing was implemented for traffic counting. Traffic is counted for every individual lane which in the end allows better interpretation of data. With this program some new possibilities of the digital image processing applications were shown.

Key words: C#, Emgu.CV, traffic counter, digital image processing

ŽIVOTOPIS

Filip Maras rođen je 03.03.1995 u Osijeku. U Čepinu završava osnovnu školu „Miroslav Krleža“ nakon koje 2010. upisuje Prvu gimnaziju u Osijeku. Od 2014. godine redovan je student na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer računarstvo.